15

University of Bahrain
College of Information Technology
Department of Computer Science
Summer Semester, 2011 – 12

ITCS215 Data Structures

Quiz 1

Name_____ **ID #**_____ **Sec**_____

Consider the following class definition:

```
class Vehicle
{
   private:
     string color;
     int yearModel; // such as 2011 model
     double price;
   public:
     void setVehicle(string, int, double);
     string getColor();
     int getYearModel();
     double getPrice();
     void print();// prints all attributes
     vehicle(string, int, double);
};
```

(A) Write a class called **Car** which inherits all the properties of class **Vehicle** with inheritance type as public. This new class will have the following additional members:

Data members (private): **typeOfFuel** (string), **sittingCapacity**(int)

Member Functions (public):

- constructor having 5 parameters to initialize all the 5 data members.
- Set and get functions for **typeOfFuel** and **sittingCapacity**.
- print(): calls the print() of the base class **Vehicle** to display the inherited data members, and also prints the 2 remaining data members of class **Car.**
- age: which returns (2012 – **yearModel).**

Write only prototypes of all member functions in the class **Car**.

(B) Write definitions (implementation) of the following member functions of class **Car**: **Constructor, print and age**.

1)

```
class Car:public Vehicle {
private:
string typeOfFuel;
int sittingCapacity;
public:
Car(string typeF, int sitC, string C, int yearM, double p);
Void setTypeOfFuel(string typeF);
Void setSittingCapacity(int sitC);
string getTypeOfFuel();
int getSittingCapacity();
Void print();
int age();
};
```

7.5

(B)

```
Car::Car(string typeF, int sitC, string C, int yearM, double p):Vehicle(C, yearM, p)
{
  typeOfFuel = typeF;
  sittingCapacity = sitC;
}
Void Car::print(){
  Vehicle::print();
cout<<"Type of fuel:"<<typeOfFuel<<endl;
cout<<"Sitting Capacity:"<<sittingCapacity<<endl;
}
int Car::age(){
 return (2012 - getYearModel());
 }
```

7.5

15

University of Bahrain
College of Information Technology
Department of Computer Science
Summer Semester, 2011 – 12

ITCS215 Data Structures

Quiz 1

Name_____ **ID #**_____ **Sec**_____

Write a function ***printCommon*** that accepts two array based lists *list1* and *list2* as parameters and prints all those elements which occurs in both lists. Assume that class ***arrayListType*** is available for use.

Function prototype:

```
template<class Type>
void printCommon(arrayListType<Type>& list1,
                 arrayListType<Type>& list2);
```

You can use the following operations of the class ***arrayListType*** in writing the required function:

isEmpty, retrieveAt, seqSearch, listSize and *print.*

```
template<class Type>
void printCommon(arrayListType<Type>& list1, arrayListType<Type>& list2){
    if( list1.isEmpty() || list2.isEmpty())
        cout<<"list is empty"<<endl;
    else{
// bool flag;
  Type v1, v2;

  for(int i=0; i< list1.listSize(); i++){
list1.retrieveAt(i, v1);
    if( list2.seqSearch(v1)!= -1)
      cout<< v1<<" ";
    }
}
```

15

University of Bahrain
College of Information Technology
Department of Computer Science
Summer Semester, 2011 – 12

ITCS215 Data Structures

Quiz 3

Name_____ **ID #**_____ **Sec**_____

Write a member function **splitNode** to be included in class `linkedListType` which accepts a parameter ***upperBound*** of type *Type*. The function will split the nodes whose *info* values are larger than ***upperBound***. The splitting is done by replacing the *info* value with half its original value and then inserting a new node with the other half value, at the end of the list. If the list is empty then return false, otherwise return true.

Function Prototype:

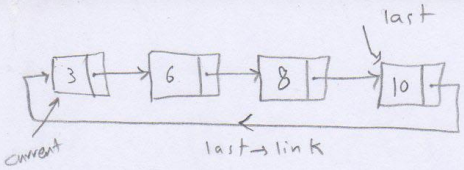
bool splitNode(const Type& upperBound);

If you call any member function of class `linkedlistType` in your member function, then write it also.

Example:
upperBound = 15
List (before function call): 4 **19** 7 9 **23** **18** 2 13

List (after function call) : 4 **9.5** 7 9 **11.5** **9** 2 13 **9.5** **11.5** **9**

```
template<class Type>
bool linkedListType<Type>::splitNode(const Type& upperBound){
 if(first==NULL){
cout<<"list is empty"<<endl;
return false;}
nodeType<Type> *current=first, *last1=last, newNode;
                int size=count;
                for(int i=0;i<size;i++){
 if(current->info > upperBound){
 current->info=(current->info)/2.0;
 newNode=new nodeType<Type>; assert(newNode!=NULL);
 newNode->info=current->info;
 last->link=newNode;
 newNode->link=NULL;
 last=newNode;
count++;}
else
current=current->link;
}
return true;
}
```



14

University of Bahrain
College of Information Technology
Department of Computer Science
Summer Semester, 2011 – 12

ITCS215 Data Structures

Quiz 4

Name _____ **ID #** _____ **Sec** ____

Write following member functions to be included in class circularLinkedList. Assume that the class contains a private data member ***last***, which points to the last node of the list.

(1) print: to print the info of all the nodes.

(2) insertLast: to insert a new node at the end of the list.

(1)

```
template<class Type>
Void circularLinkedList<Type>::print(){
    if (last == NULL) {
    Cout<<"list is empty"<<endl;
    return; }
NodeType<Type> *current = last->link;
do{
  cout<< current->info <<" ";
  current= current->link; }
while (current != last->link);
}
```

7.5

(2)

```
template <class Type>
void circularLinkedList<Type>::insertLast(Type& item)
{
 nodeType<Type> *newNode;
 newNode=new nodeType<Type>;
assert(newNode != NULL);
 newNode->info=item;
newNode->link=NULL;
if (last==NULL){
newNode->link= newNode;
last= newNode;
 }
else{
newNode->link= last->link;
last= newNode;
}
}
```

6.5

last->link = newNode;